"Express Mail" Mailing Label No. **EL436468003US**

PATENT APPLICATION
ATTORNEY DOCKET NO. OR00-03802

INS. B1

METHOD AND APPARATUS FOR AUTOMATIC DATABASE ENCRYPTION

Inventor: Richard R. Wessman

INS. B2

~~BACKGROUND~~

Field of the Invention

The present invention relates to computer security and databases within computer systems. More specifically, the present invention relates to a method and apparatus for automatically encrypting and decrypting data to be stored in a database.

Related Art

Modern database systems store and retrieve vast quantities of information. Some of this information is sensitive, such as credit card numbers, bank balances, and nuclear secrets, and hence must be protected so that the information does not end up in the wrong hands.

Some database systems are able to restrict access to specific information by using access controls that are specified in security profiles assigned to each client. Such systems prevent a client from accessing information other than what has been authorized for the client. This normally protects the sensitive information and, therefore, leads users to trust the database system to ensure that information stored within the database system remains secret.

There is, however, a major weakness in these types of database systems. The data base administrator (DBA) has access to everything that is stored within the database system. This unrestricted access allows an unscrupulous DBA to steal information from the database system and to use the stolen information for illicit purposes. Note that is not practical to implement access controls for the DBA because doing so prevents the DBA from performing necessary database maintenance functions.

Sensitive information can be kept secret from the DBA by encrypting the sensitive information within the user application at the client. In this approach, all sensitive information is stored in an encrypted form within the database system and is consequently protected from examination by the DBA. This approach has the advantage that the DBA is not restricted from performing database maintenance functions. A major drawback to this approach, however, is that all user applications that handle sensitive information need to be able to encrypt and decrypt information. Providing such encryption and decryption code in all of the numerous applications that handle sensitive data is very inefficient.

What is needed is a method and an apparatus that allows a DBA to have unrestricted access to the database system while protecting sensitive information within the database system in an efficient manner.

INS. B3 ~~SUMMARY~~

One embodiment of the present invention provides a system for managing encryption within a database system that is managed by a database administrator, and wherein a user administrator not otherwise associated with the database system, manages users of the database system. This system performs encryption automatically and transparently to a user of the database system. The system operates by receiving a request to store data in a column of the database system. If a user has designated the column as an encrypted column, the system automatically encrypts the data using an encryption function. This encryption function uses a key stored in a keyfile managed by the security administrator. After encrypting the data, the system stores the data in the database system using a storage function of the database system.

In one embodiment of the present invention, the system manages decrypting encrypted data stored in the database system. The system operates by receiving a request to retrieve data from the encrypted column of the database system. If the request to retrieve data is from an authorized user of the database system, the system allows the authorized user to decrypt encrypted data, otherwise, the system prevents decrypting encrypted data if the request to retrieve data is received from the database administrator, the security administrator, or the user administrator.

In one embodiment of the present invention, the security administrator selects the mode of encryption for the column. The mode of encryption can be, but is not limited to, data encryption standard (DES) or triple DES.

In one embodiment of the present invention, the security administrator, the database administrator, and the user administrator are distinct roles. A person selected for one of these roles is not allowed to be selected for another of these roles.

In one embodiment of the present invention, the security administrator manages the keyfile. In doing so, the security administrator creates the keyfile. Next, the security administrator establishes how many keys are to be stored in the keyfile. The security administrator then establishes a relationship between a key identifier and the key stored in the keyfile. The keyfile can be stored in a location such as an encrypted file in the database system, or a location separate from the database system. Finally, the security administrator moves an obfuscated copy of the keyfile to a volatile memory within a server associated with the database system.

In one embodiment of the present invention, the security administrator specifies a column to be encrypted. If the column currently contains encrypted data, the system decrypts the data using the previous key. After decrypting the encrypted data or if the column contains clear-text data, the system encrypts the data using a new key.

In one embodiment of the present invention, the key identifier associated with the encrypted column is stored as metadata associated with a table containing the encrypted column within the database system.

In one embodiment of the present invention, the security administrator establishes encryption parameters for the encrypted column. These encryption parameters include, but are not limited to, encryption mode, key length, and integrity type. The security administrator can manually enter the encryption parameters for an encrypted column. The security administrator can also establish a profile table in the database system for saving and recovering encryption parameters for the encrypted column.

INS. 34 ~~BRIEF DESCRIPTION OF THE FIGURES~~

FIG. 1 illustrates a database system in accordance with an embodiment of the present invention.

FIG. 2 illustrates details of a database system in accordance with an embodiment of the present invention.

FIG. 3 is a flowchart illustrating the process of creating a keyfile in accordance with an embodiment of the present invention.

FIG. 4 is a flowchart illustrating the process of creating an encryption profile in accordance with an embodiment of the present invention.

FIG. 5 is a flowchart illustrating the process of establishing a column in the database as an encrypted column in accordance with an embodiment of the present invention.

FIG. 6 is a flowchart illustrating the process of storing data in the database system in accordance with an embodiment of the present invention.

FIG. 7 is a flowchart illustrating the process of retrieving data from the database system in accordance with an embodiment of the present invention.

INS. 35 ~~DETAILED DESCRIPTION~~

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Database System

FIG. 1 illustrates a database system in accordance with an embodiment of the present invention. As illustrated in FIG. 1, client 110 is coupled to database server 112. Client 110 and database server 112 may include any type of computer system, including, but not limited to, a computer system based on a microprocessor, a mainframe computer, a digital signal processor, a personal organizer, a device controller, and a computational engine within an appliance.

Database server 112 is also coupled to database 118. Database 118 can include any type of system for storing data in non-volatile storage. This includes, but is not limited to, systems based upon magnetic, optical, and magneto-optical storage devices, as well as storage devices based on flash memory and/or battery-backed up memory.

Database server 112 includes key management function 114 and obfuscated keyfile 116. Obfuscated keyfile 116 contains a copy of the data in keyfile 120. Keyfile 120 contains keys and key identifiers for encrypting and decrypting data. Keyfile 120 is stored on a system separate from the database system or can be stored as an encrypted table in database 118.

User 102 accesses database 118 through client 110. User administrator 104 grants privileges to user 102 for accessing database 118. User administrator 104 is not allowed to access the database.

Security administrator 106 manages the encryption system through database server 112. Managing the encryption system includes, but is not limited to managing keyfile 120 and specifying which columns of tables in database 118 are encrypted.

Database administrator 108, manages the database system by performing services such as data backup, data recovery, storage allocation, and the like.

Within the database system, user administrator 104, security administrator 106, and database administrator 108 are distinct roles. A person selected for any one of these roles may not be selected to perform any of the other roles.

Database Details

FIG. 2 illustrates details of a database system in accordance with an embodiment of the present invention. In addition to key management function 114 and obfuscated keyfile 116, database server 112 also includes, but is not limited to, encryption function 204, decryption function 206, storing function 208, and retrieving function 210.

Encryption function 204 uses keys from obfuscated keyfile 116 to encrypt data 202 received from client 110. Decryption function 206 uses keys from obfuscated keyfile 116 to decrypt data 212 received from database 118. Storing function 208 stores data 212 in database 118, while retrieving function 210 retrieves data 212 from database 118.

Database 118 includes, but is not limited to, table 218, profiles 220, and metadata 222. Table 218 is organized with related data located in a single row that spans columns 224, 226, 228, and 230. As illustrated in FIG. 2, the first row

of table 218 contains XXX in column 224, national identifier NID in column 226, YYY in column 228, and ZZZ in column 230. Data 212 is stored and retrieved from rows of table 218.

Metadata 222 stores encryption parameters for table 218 in columns 240, 242, 244, and 246. The first row of metadata 222 relates to column 226 in table 218. Column 240 identifies column 226 as encrypted. Column 242 contains the key identifier for the key within obfuscated keyfile 116 that is used to encrypt data in column 226. Column 244 indicates the encryption mode. In this example, data encryption standard (DES) encryption is being used. Column 246 indicates the data integrity mechanism to be used to detect tampering with the encrypted data. In this example, message digest 5 (MD5) is being used. Encryption parameters are supplied to database server 112 as parameters 216 for storing in metadata 222.

Profiles 220 stores encryption profiles created by security administrator 106 in columns 232, 234, 236, and 238. The first row of profiles 220 contains a profile. Column 232 indicates the name, 999, of the profile. Column 234 indicates the encryption mode. In this example, the profile indicates that data encryption standard (DES) encryption is being used. Column 236 indicates the key-length to use. This example indicates a key-length of 56 bits. Column 238 indicates the data integrity mechanism to be used with the profile. This example indicates that secure hash algorithm 1 (SHA-1) is being used. Profiles are supplied to database 118 as profile 214.

Creating a Keyfile

FIG. 3 is a flowchart illustrating the process of creating keyfile 120 in accordance with an embodiment of the present invention. The system starts when key management function 114 receives a request from security administrator 106 to create keyfile 120 (step 302). Key management function 114 receives the

number of keys to create from security administrator 106 (step 304). Next, key management function 114 receives the name of keyfile 120 from security administrator 106 (step 306). Key management function 114 also receives a random key generator seed from security administrator 106 (step 308).

Key management function 114 generates the keys and matching key identifiers (step 310). Next, key management function 114 stores keyfile 120 (step 312). Note that keyfile 120 may be stored in a location remote to the database system or may be stored as an encrypted table within database 118.

Finally, key management function 114 makes an obfuscated copy of keyfile 120 and stores it as obfuscated keyfile 116 in volatile memory of database server 112 (step 314).

Creating a Profile

FIG. 4 is a flowchart illustrating the process of creating an encryption profile in accordance with an embodiment of the present invention. The system starts when key management function 114 receives a request from security administrator 106 to create an encryption profile (step 402). Key management function 114 receives the name of the profile to create from security administrator 106 (step 404). Next, key management function 114 receives the encryption algorithm to associate with the profile (step 406). Key management function 114 then receives the key-length to associate with the profile (step 408). Next, key management function 114 receives the type of data integrity to associate with the profile (step 410). Key management function 114 creates the profile (step 412). Finally, key management function 114 stores the profile, consisting of the profile name, encryption mode, key-length, and integrity type in columns 232, 234, 236, and 238, respectively, in the next available row of profiles 220 (step 414).

Establishing an Encrypted Column

FIG. 5 is a flowchart illustrating the process of establishing a column in the database as an encrypted column in accordance with an embodiment of the present invention. The system starts when database server 112 receives a request to encrypt a column, say column 226, of table 218 in database 118 (step 502). Database server 112 first determines how security administrator 106 specified the encryption parameters (step 504).

If the encryption parameters are supplied by using a profile, database server 112 retrieves the profile 214 from profiles 220 in database 118 (step 506). After retrieving the encryption parameters form profile 214 or if the parameters were supplied in the request at step 504, database server 112 determines if the column already contains data (step 508).

If the column already contains data in step 508, database server 112 inspects metadata 222 to determine if the data in the column was previously encrypted (step 510). If the data in the column was previously encrypted in step 510, retrieving function 210 retrieves the cipher-text data from table 218 (step 512). Next, decryption function 206 decrypts the data using the previous key obtained from metadata 222 (step 514).

If the data is not encrypted at step 510, retrieving function 210 retrieves the clear-text data from table 218 (step 516). When the clear-text is available after step 514, or step 516, encryption function 204 encrypts the data (step 518). Next, storing function 208 stores the cipher-text data in table 218 (step 520).

If the column does not contain data at step 508 or after the cipher-text data is stored in step 520, database server 112 stores the encryption parameters for the column in metadata 222 (step 522).

Storing Data in the Database

FIG. 6 is a flowchart illustrating the process of storing data in database 118 in accordance with an embodiment of the present invention. The system starts when database server 112 receives a request to store data 202 from client 110 (step 602). Database server 112 examines metadata 222 to determine if the column where the data will be stored is encrypted (step 604). If the column is encrypted (step 606), database server 112 retrieves the encryption parameters for the column from metadata 222 (step 608). Database server 112 then retrieves the encryption key related to the key identifier (KID) from obfuscated keyfile 116 (step 609). Next, encryption function 204 encrypts the data (step 610). After the data is encrypted in step 610 or if the column is not encrypted at step 606, storing function 208 stores the data in table 218 (step 612).

Retrieving Data from the Database

FIG. 7 is a flowchart illustrating the process of retrieving data from database 118 in accordance with an embodiment of the present invention. The system starts when database server 112 receives a request from client 110 to retrieve data from database 118 (step 702). Retrieving function 210 retrieves the data from table 218 in database 118 (step 704). Next, database server 112 determines if the request is from an authorized user (step 709).

If the request is from an authorized user at step 709, database server 112 examines metadata 222 to determine if the column related to the data is encrypted (step 708). If database server 112 determines that the data is encrypted in step 708, database server 112 retrieves the encryption parameters from metadata 222 (step 710). Database server uses the key identifier (KID) to retrieve the decryption key from obfuscated keyfile 116.

Next, decryption function 206 decrypts the data (step 712). After the data is decrypted in step 712 or if the data was determined to not be encrypted in step 708, database server 112 returns the data to client 110 (step 714). If the request is not from an authorized user at step 709, the data is not returned to the client. Specifically, the database administrator, the security administrator, and the user administrator are not authorized users and, therefore, are prevented from decrypting and receiving encrypted data stored within the database.

The foregoing descriptions of embodiments of the invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.